Amendments to Claims

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims

1. (currently amended) A system of managing data utilizing one or more processors and a single operating system in a multi-thread environment, comprising:

a plurality of map components, each map component having one or more ports for accepting data and for producing data and each map component encapsulating a particular dataflow pattern;

compiler tools for organizing and linking said map components using said ports into ~~an executable~~ a dataflow application, said compiler tools, including a map synthesizer which validates links between map components and prepares an execution plan; and

an executor for creating and managing data communication among map components in the dataflow application and executing the dataflow application on said one or more processors in parallel using said execution plan, wherein said executor assigns a separate thread to map processes ~~with each map component~~ as a separate thread of execution with data supplied to the system.

2. (currently amended) The system of claim 1, the compiler ~~including~~ tools ~~for~~ include a tool for visually creating composite components comprising other map components and ~~tools~~ a tool for visually assembling map components into a dataflow application.

3. (original) The system of claim 1, at least one map component having properties determining map component design behavior.

4. (original) The system of claim 1, at least one map component having properties that affect map component execution behavior.

5. (original) The system of claim 1, at least one of the map components comprising a composite component encapsulating a particular dataflow pattern using other map components as subcomponents.

6. (original) The system of claim 1, at least one of the map components comprising a scalar map component to process a specific data transformation.

7. (original) The system of claim 1, at least one of said ports linked to transfer specific types of data.

8. (original) The system of claim 1, at least one of said ports initially defined as a generic port for processing generic types of data, said generic port being later synthesized to transfer a specific sub-type of data.

9. (original) The system of claim 1, at least one of said ports being composite, comprising a plurality of hierarchical ports.

10. (original) The system of claim 1, at least one of said ports supporting multi-valued null data tokens.

11. (previously presented) The system of claim 1, at least one of said map components being encoded as an encrypted extensible markup language (XML) document.

12. (original) The system of claim 1, at least one of said map components being composite comprising a number of hierarchical dataflow graphs.

13. (currently amended) The system of claim 1, the compiler tools operating to remove design time links between map components to produce a flat dataflow graph containing a plurality of map processes for execution.

14. (canceled)

15. (currently amended) The system of claim 1, the ~~compiler tools~~ synthesizer operating to perform syntactic and semantic analysis, type inference and validation.

16. (currently amended) A method of transforming data in a multi-thread parallel processing environment comprising a single operating system and one or more processors wherein:

map components are assembled visually into an integrated dataflow application ~~by~~ using compiler tools linking map processes within map components, wherein the map components are linked using ports;

the integrated dataflow application is executed in parallel by recognizing the linked map processes within the map components and allocating a separate thread to each of a plurality of linked map ~~process~~ processes; and

each of said plurality of linked map process is executed on its allocated thread substantially in parallel as part of a single process, and said data resides in memory accessible to each of said plurality of linked map process.

17. (original) The method of claim 16, wherein said ~~a~~ plurality of linked map processes read data tokens from input ports and write data tokens to output ports.

18. (currently amended) A method of managing data in a multi-thread environment, comprising:

accessing a library of map components at least some of said map components constituting a specific data transformation and having input and output ports;

assembling a dataflow application using map components from said library, wherein said map components each comprise one or more processes, and compiler tools ~~linked~~ link map components together using said ports; and

executing the assembled dataflow application with source data by assigning a separate thread to ~~each~~ a separate map ~~component~~ process where said threads execute as part of a single process ~~in parallel~~ on said source data without data partitioning.

19. (original) The method of claim 18, including imposing properties on the map components during assembly constraining the assemblage of the dataflow application.

20. (original) The method of claim 18, the map components including polymorphic ports which declare status as input and output ports during assemblage.

21. (previously presented) The system of claim 14, the executor operating on a single CPU in a hyperthread architecture.

22. (previously presented) The system of claim 14, the executor operating on a multiple processor core with at least some threads assigned to different processors.

23. (previously presented) The system of claim 14, the executor operating on multiple processors in a distributed network configuration.

24. (previously presented) The method of claim 16, communication between said processes executing in parallel being managed by an executor separate from the operating system.

25. (previously presented) The method of claim 18, including:

determining if a port will block execution of a thread; and

avoiding a deadlock by allowing the data queue to grow at said determined port.

26. (new) The method of claim 16, at least one map component being composite, thereby encapsulating a particular dataflow pattern.